
SoCal
NOvA in the North
2006

Caltech Effort

People

1. **Jason Trevor:** Light output test for NOvA
2. **Hai Zheng:** ν_e reconstruction for MINOS
3. **Caius Howcroft**
4. **One future grad student.**

Objectives

1. **PHYSICS:**
   1. Do realistic study of NOvA sensitivity
   2. Investigate some proposed improvements for background separation
      1. Gap finding to identify pi_zeros, what about shower charge fluctuations?
      2. P_T measurements to getter better cosmic ray separation.
      3. PID from dE/dx measurements
2. But....
   1. There are no complete framework, nothing in a modern language.
   2. No detector simulation an No attempt to match the event simulations to scintillator tests.
   3. No decent pattern recognition software.

Want physics results by yesterday, need tools now -> SoCal Framework

SoCal Framework

1. To do any physics, we needed a usable framework to do it in... so we wrote one.

Philosophy

1. **Simple:** There is no C++ magic (or frills) going on. Clear owner ship rules. Clear object lifetime. Maximal use of industry standard tools, e.g. standard rather than home grown memory management, stl iterators
    1. May be slightly easier to make mistakes (memory management)
    2. But much much easier to find problems when they do happen.
    3. **and...** the lack of "hand holding" restrictions means the framework is more flexible.
2. **Fast**: miminal use of inheritance, keep I/O speed and size at the fore front of the mind. Taken lesson learned in MINOS to heart.
3. Get writing useful code **NOW**.
    1. It is expected that it will be replaced with a full framework, but want to starting thinking about physics now.

Constrictions:

1. C++: It's a crazy to give a load of physicists.
2. ROOT: ...

Structure

A collection of libraries to allow the user to write their own programs.

SoCal Core Libraries:

1. Data format and storage: A way to store and pass around data and reconstruction objects between algorithms. Way to store data and reco.
2. Geometry: Algorithms (and users) need to know where strips/planes etc.. are. This should be easy to use for all 3 detectors.
3. ConnectionMap: Know what optical channels relate to what readout.
4. Units/Conventions: Useful functions for the user
5. Event Display:

We are now working on some user libraries to do physics:

User Libraries:

1. Photon Transport: Converts energy deposition into photons at the APD.
2. Readout Simulation: Simulates the readout chain, and produces simulated "Raw Data"
3. Reco Event: a nue reconstruction package.

Data Format: Lesson learned

1. MINOS has two data i/o formats:
   1. Candidates: Fully fledged Object IO, with lots of auto-pointers, full object trees. (Based on BaBar system)
   2. Ntuples: The Candidates collapsed down to flat objects. Store approximately the same amount of information.
2. While candidates have two advantages:
   1. As they are the objects used in memory, they can be reread back into the same framework.
   2. They use pointers. E.g. Easy to get hits in a track.
   3. They automatically delete objects trough a “handle” system

3. They have some draw backs:
   1. They are large
   2. They are slow (5 times slower than ntuples)
   3. The auto-delete feature is often confusing rather than a help.
   4. They are very sensitive to version sheer.

Raw Data	77GB
Candidate	298 GB
Ntuple	49 GB

SoCal Data Format

```
namespace SoCal{
  class Record
  {
  public:
    Record();
    virtual ~Record();
    void Reset();
    void Print();
    EventId                 dId;
    TimeStamp               dTime;
    MCInfo                  dMC;
    std::vector<Digit>      dDigits;
    std::vector<TObject*>   dUserData;
  };

}
```

```
namespace SoCal{
  class MCInfo{
  public:
    MCInfo();
    virtual ~MCInfo();
    void Reset();
    std::vector<NeuKin>       dInteractions;
    std::vector<StdHEP>       dParticles;
    std::vector<ScintHit>     dHits;
    std::vector<PESignal>     dPESignals;
    void Print();    //Debug stuff
  };
}
```

Bin Area: Users can dump reco objects in here for IO

Geometry + ChannelMap

Geom User Interface:

1. Users get a specific geometry from the GeometryStore, several geometries can exist at once.
2. Each Geometry has a list of objects for each plane, use these to find out where strips in each plane are, their size, fiber locations etc...

Geom Storage and Building

1. Geometries are constructed from:
   1. REROOT files (the output of the fortran simulation) - implemeted
   2. ROOT file. Can write out a file with your reco data.

Geom Tested with..

1. CDR geometry
2. TAVC geometry
3. Near Det geometry

Channel Map:

1. Holds information about how to map optical channels to readout channels
2. Currently a place holder, actual electronics data channel number is encapsulated in the SoCal::ChannelId class.

Event Display

Event Source

Branding

Reco'ed event

Decay chain

Colour = energy deposited

e/μ

π

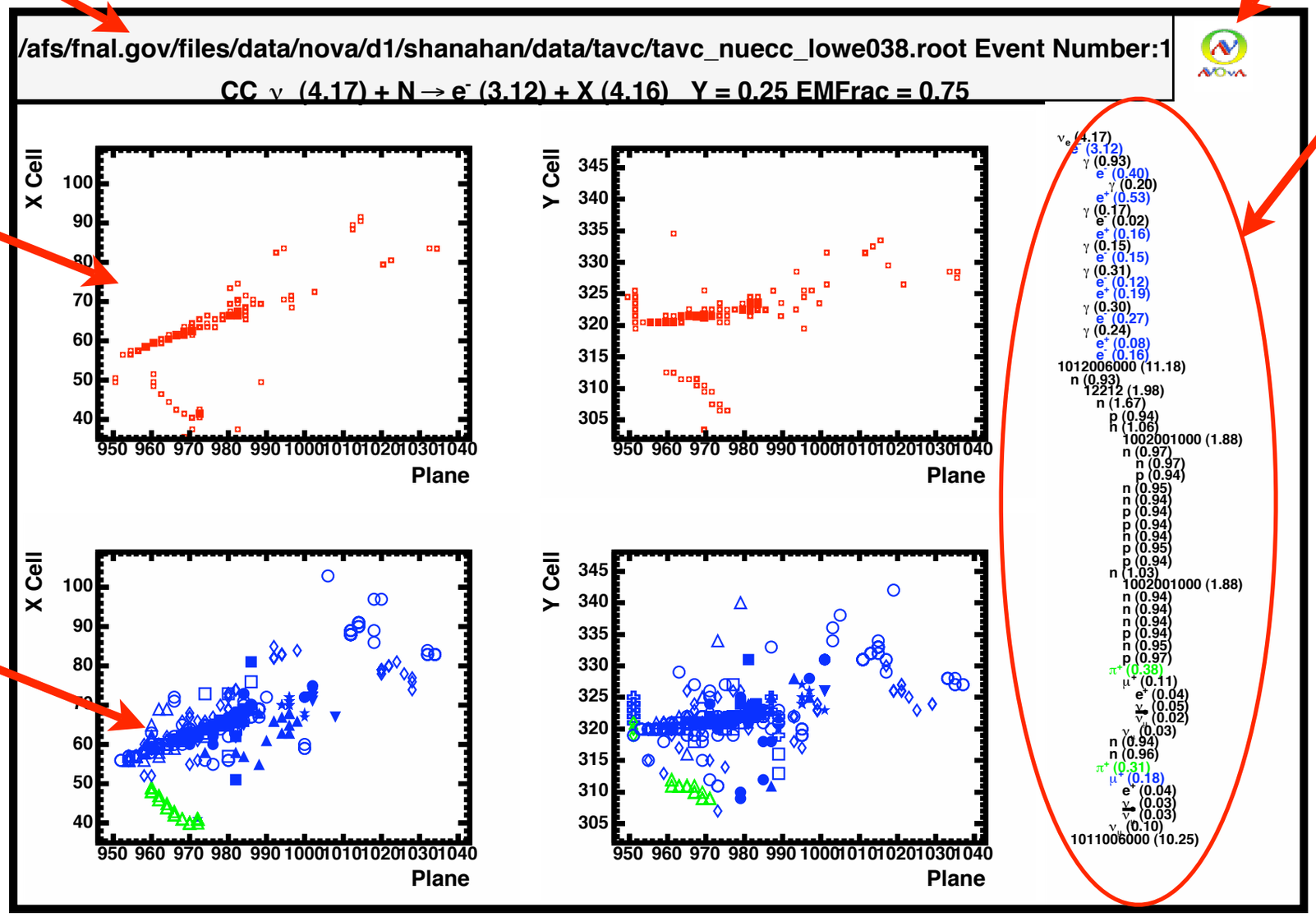
p

True Hits

e/μ

π

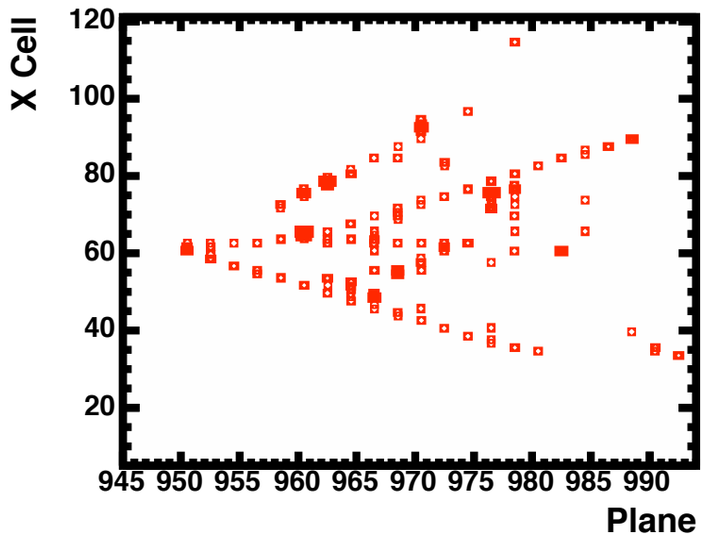
p

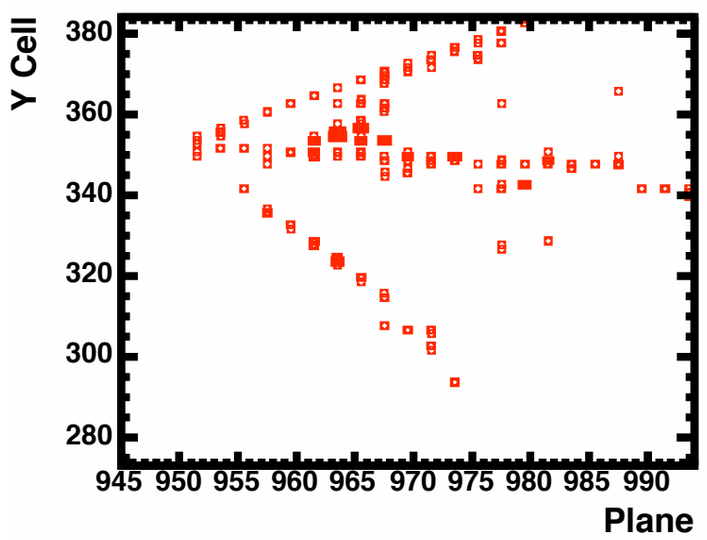


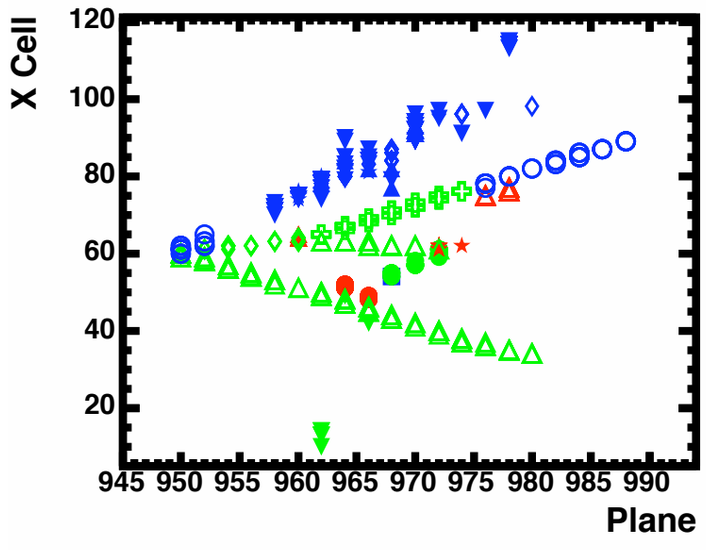
E.G. 2

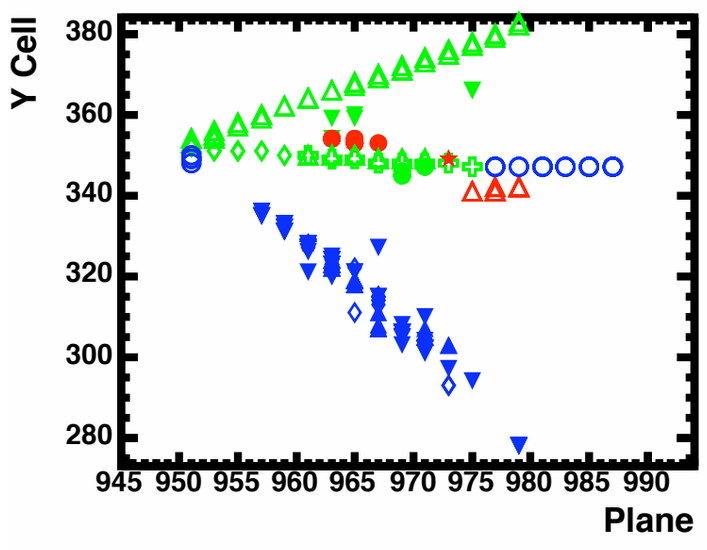
/afs/fnal.gov/files/data/nova/d1/shanahan/data/tavc/tavc_nuecc_lowe038.root Event Number:4

CC ν (5.75) + N → e^- (0.91) + X (5.73) Y = 0.87 EMFrac = 0.16







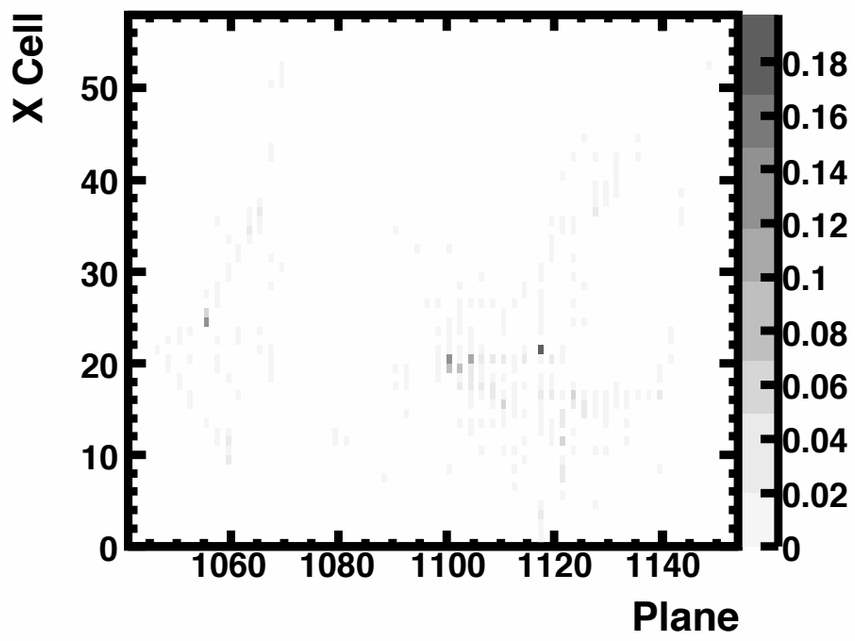


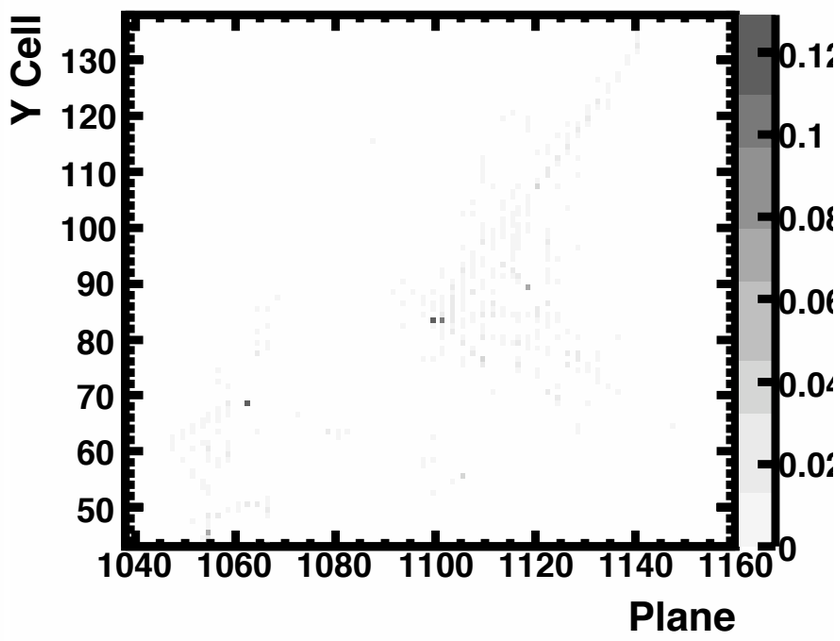
ν_e (5.75)
  e^- (0.91)
    γ (0.84)
      e^- (0.01)
      e^+ (0.83)
        γ (0.24)
          e^+ (0.03)
          e^- (0.21)
            γ (0.16)
              e^- (0.01)
              e^+ (0.15)
        γ (0.18)
          e^- (0.03)
          e^+ (0.16)
1012006000 (11.18)
  p (0.92)
    10000223 (5.76)
      π^+ (0.87)
        μ^+ (0.27)
          e^+ (0.04)
          ν_e (0.02)
          $\bar{\nu}_e$ (0.04)
          ν_μ (0.09)
      π^+ (2.49)
        1003001000 (2.81)
      π^+ (0.58)
        p (1.07)
        π^+ (0.31)
          μ^+ (0.11)
            e^+ (0.02)
            ν_e (0.04)
            $\bar{\nu}_\mu$ (0.04)
            ν_μ (0.03)
n (1.09)
  1002001000 (1.88)
    n (0.99)
    p (0.95)
    p (0.95)
    n (0.94)
    p (0.95)
    n (0.94)
    n (0.95)
    p (0.94)
    n (0.94)
π^+ (0.65)
  μ^+ (0.33)
    e^+ (0.02)
    ν_e (0.05)
    $\bar{\nu}_\mu$ (0.03)
    ν_μ (0.10)
π^0 (0.25)
  γ (0.09)
  γ (0.16)
n (1.29)
  n (1.05)
    n (0.99)
      n (0.97)
        n (0.96)
          γ (0.01)
        n (0.94)
    p (1.00)
p (1.18)
  n (0.99)
    1003001000 (2.81)

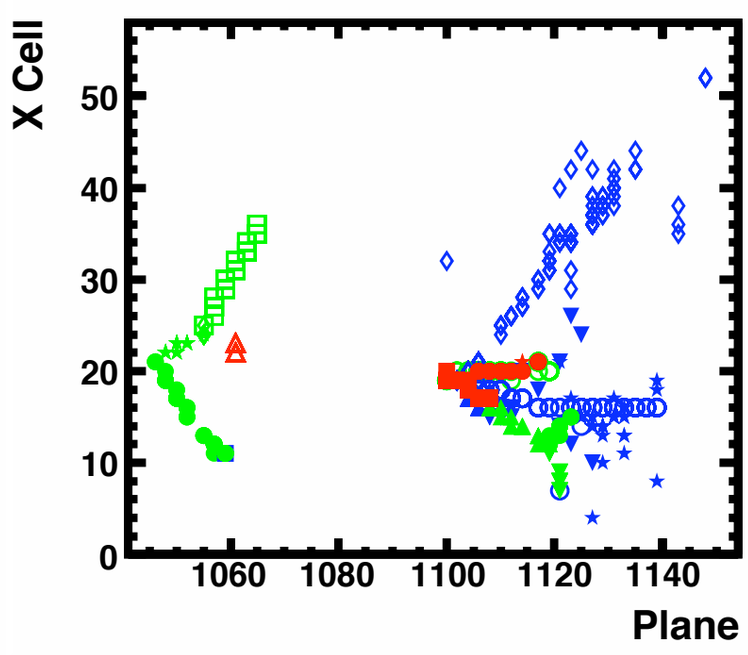
Example 3. High E NC

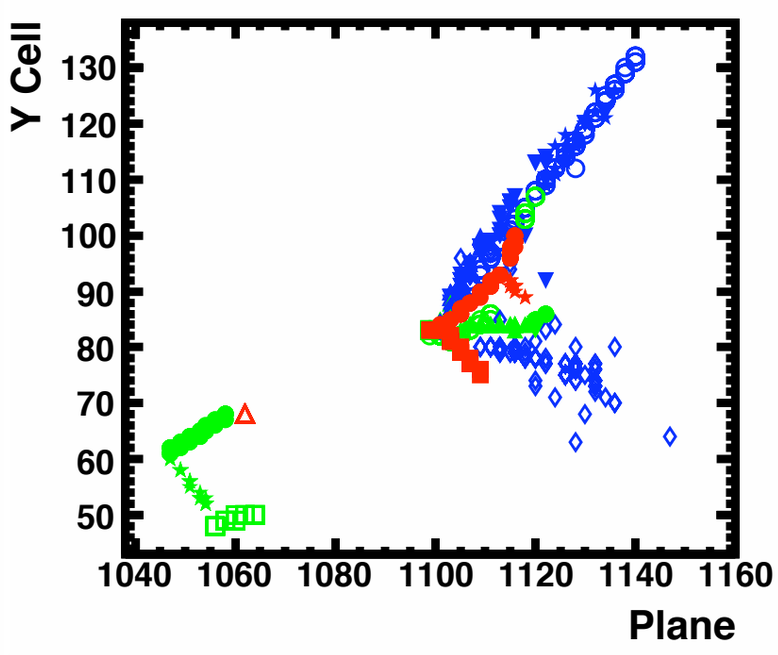
../../../../../Data/CDR_Geom/cdrc_far_numunc_highE_1.root Event Number:13

NC ν (17.56) + N $\rightarrow \nu$ + X (8.62) Y = 0.50 EMFrac = 0.00



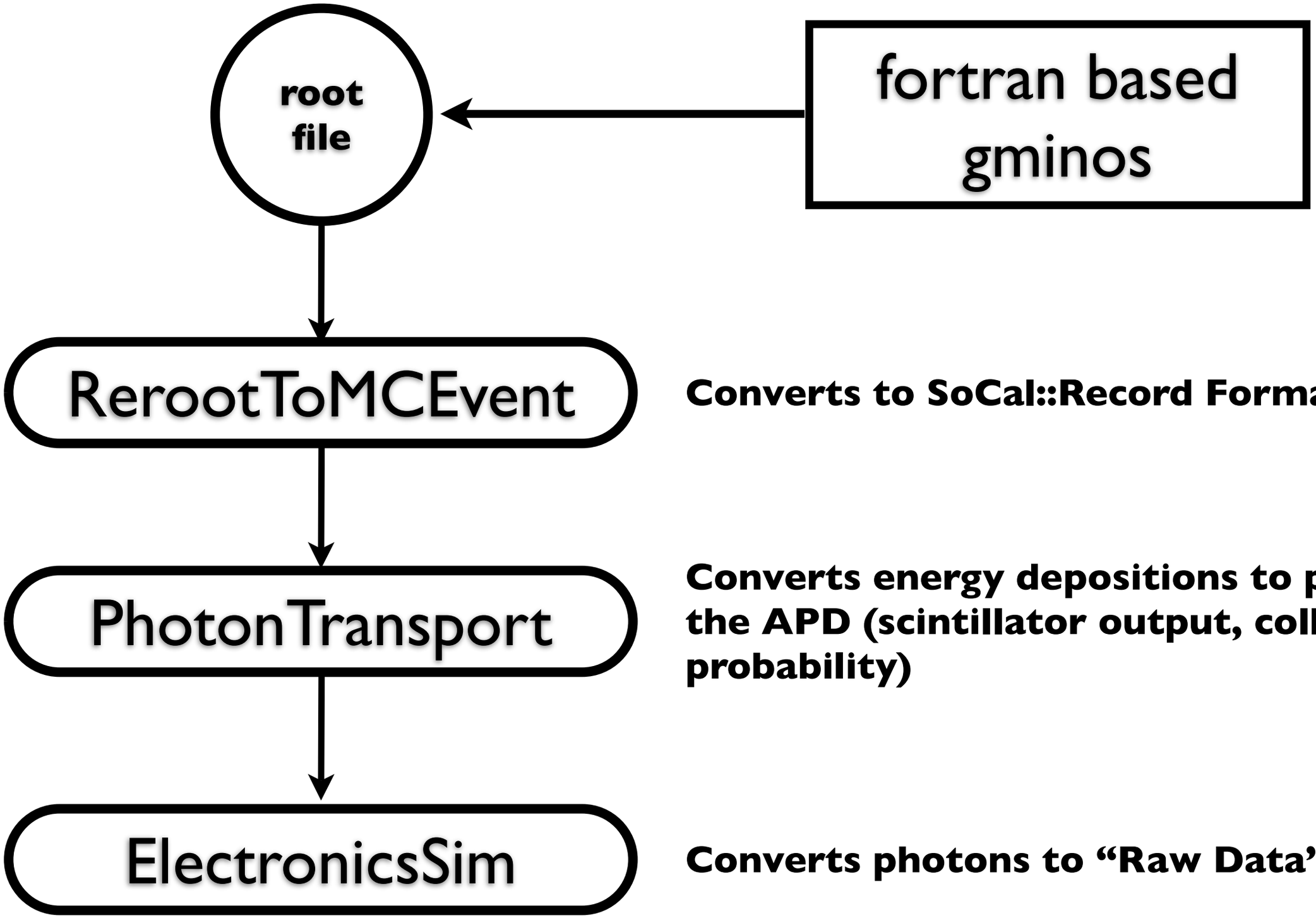






ν_μ (17.56)
ν_μ (8.93)
1012006000 (11.18)
n (0.93)
10000221 (9.56)
π^- (0.96)
π^- (0.96)
1002001000 (1.90)
π^- (0.58)
1002001000 (1.88)
π^- (0.35)
γ (0.07)
γ (0.07)
n (1.07)
1003001000 (2.81)
n (0.95)
n (0.95)
n (0.94)
p (0.95)
p (0.96)
p (1.00)
p (0.94)
p (0.95)
p (0.95)
p (0.95)
n (0.95)
n (0.94)
p (0.95)
1002001000 (1.88)
n (1.04)
1003001000 (2.81)
n (0.95)
n (0.94)
p (0.95)
p (0.95)
n (0.95)
n (0.95)
n (0.94)
p (0.95)
p (0.94)
p (0.94)
n (0.94)
n (0.94)
n (0.94)
p (1.03)
π^+ (0.39)
π^+ (0.39)
μ^+ (0.11)
e^+ (0.04)
ν_e (0.04)
$\bar{\nu}_\mu$ (0.02)
ν_μ (0.03)
n (8.21)
π^0 (0.18)
γ (0.06)
γ (0.12)
n (7.87)
p (0.98)
n (7.83)
p (1.04)
p (4.00)
1002001000 (2.00)
p (1.98)
p (1.18)

Data Flow


root file
fortran based gminos
RerootToMCEvent
Converts to SoCal::Record Format
PhotonTransport
Converts energy depositions to photons a the APD (scintillator output, collection probability)
ElectronicsSim
Converts photons to “Raw Data”.

PhotonTransport

Two versions

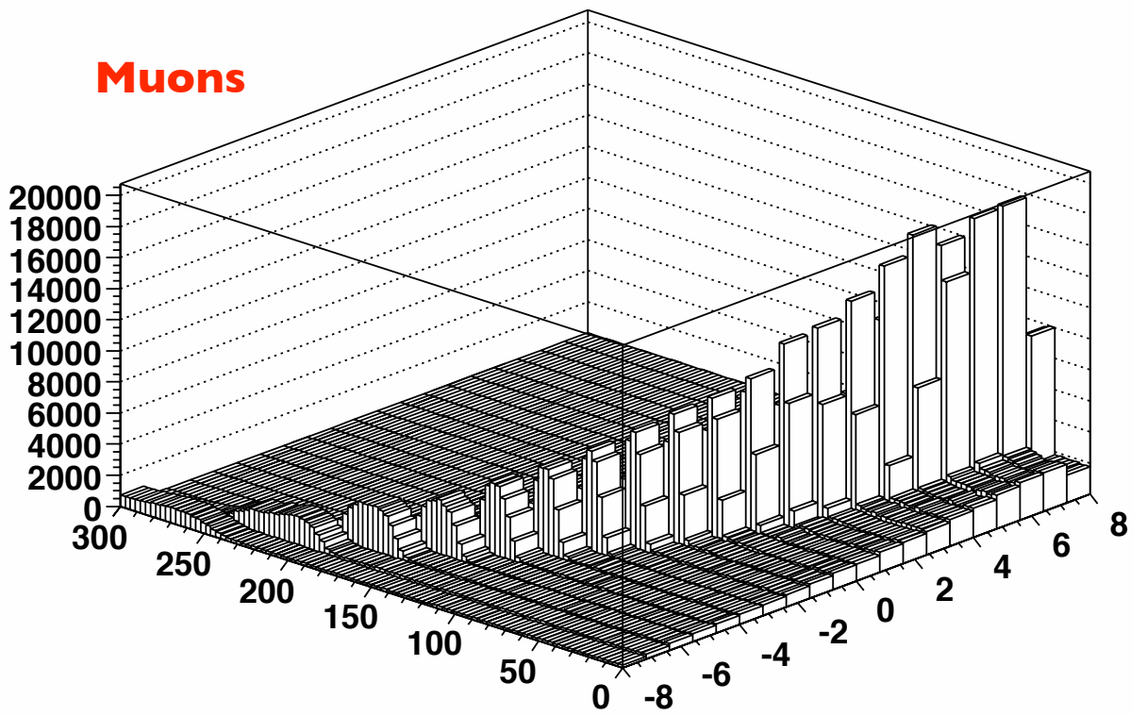
1. "Dumb" photon transport:
   1. Simple factor for scintillator output*WLS capture efficency
   2. Hard coded WLS fiber attenuation (currently minos).
   3. Hard coded quantum efficiency
   4. Tuned to CDR results (probably wrong)
   5. Is available for users.
2. "Simple" photon transport in the works:
   1. Individual photon creation and tracking, including proper treatment of production/absorption spectrum.
   2. Timescale ~months.

Plan:

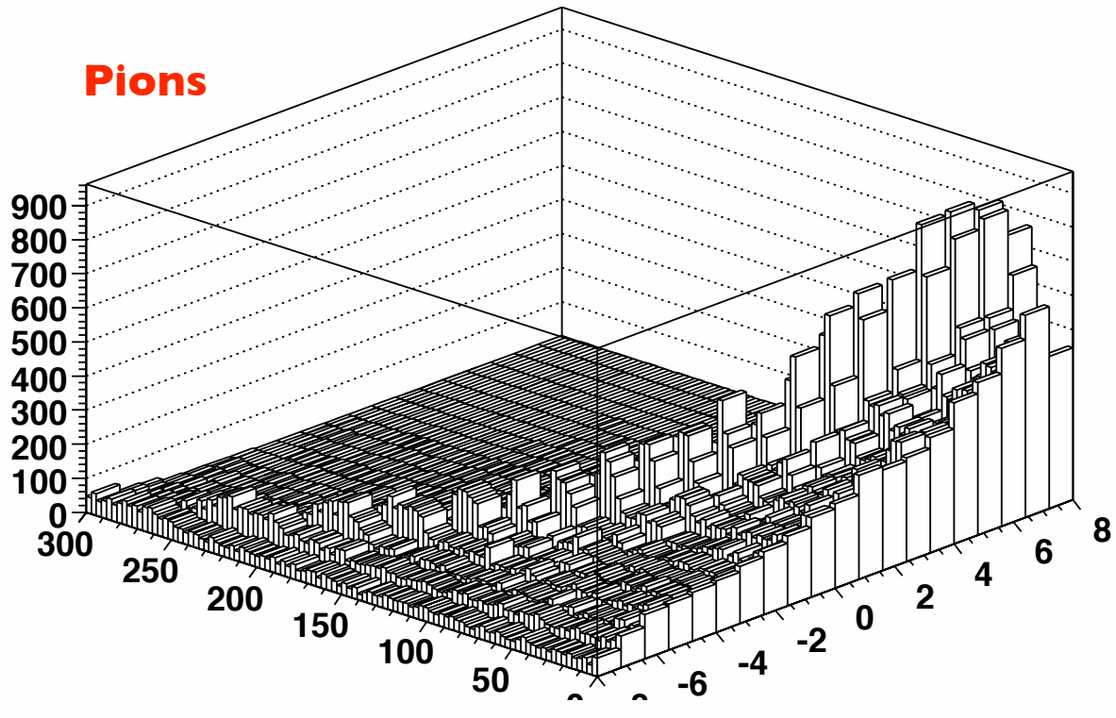
1. Caltech (Jason + minions) are doing light output tests for NOvA cells.
2. Using his data we can tune the "Simple" photon transport
3. Use this to tune a fast implementation (probably some version of "dumb").
4. Iterate.

Example

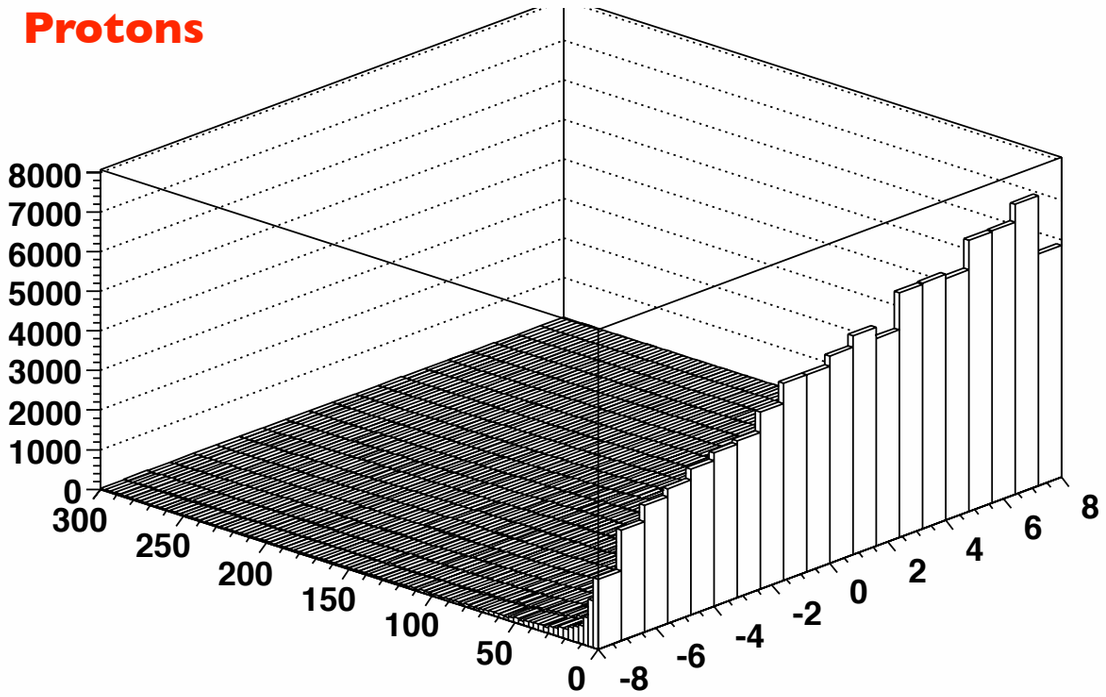
Muons



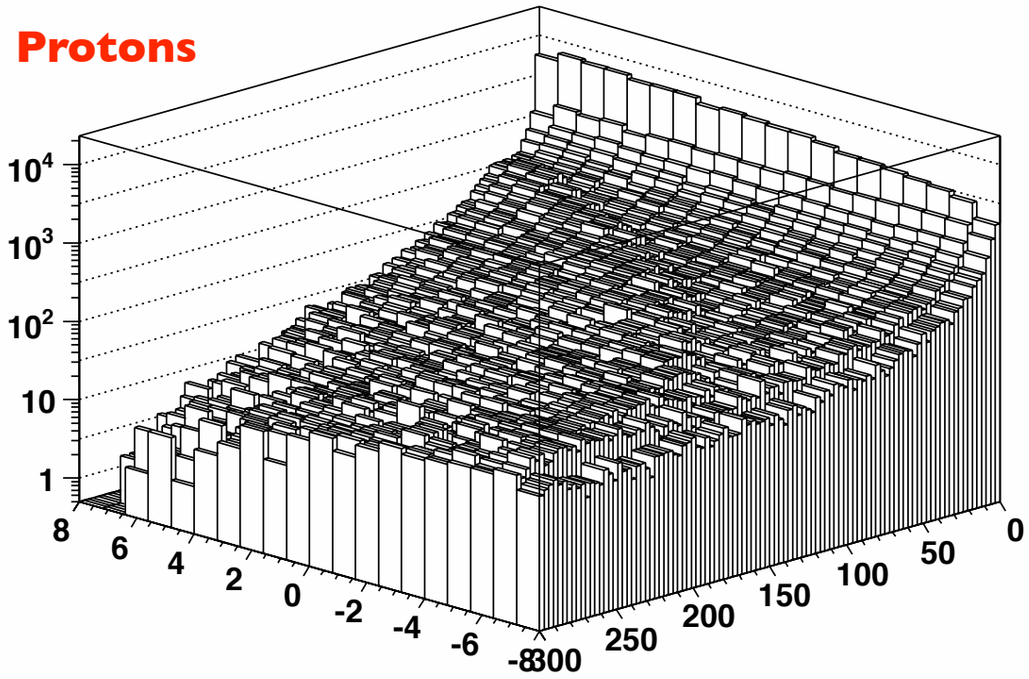
Pions



Protons



Protons



PhotonTransport Output

- Outputs SoCal::PESignals.
- Each PESignal
- One-to-one mapping between SoCal::ScintHits and SoCal::PESignals,Different to MINOS becuase:
  - NOvA has a lot more photons (memory hog)
  - The scintilltor decay time is small compared to NOvA timing resolution.
- All the PESignals in one cell are then fed into ReadoutSim to create SoCal::Digits (Raw Data)

```
namespace SoCal{
  class PESignal{
  public:
    PESignal();
    PESignal(float time, float timerms, long scinthit, int npe, const ChannelId&
 channel);
    virtual ~PESignal();
    float dTimeMean;
    float dTimeRMS;
    long  dScintHitIdx;
    int   dNPE;
    ChannelId   dChannel;
#ifdef USEROOTIO
    ClassDef(PESignal, 1);
#endif //USEROOTIO


  };
}
```

Readout Simulation

- Currently there is an implementation, called “DumbReadoutSim”.
- Threshold set set by default to 12 pes.
- Does simulate saturated electronics
- Hard coded PE/ADC conversion of ~4 ADCs/PE.
- Timing resolution is 500 ns bins. No fancy stuff done yet.
- No amp. noise.

Currently very silly (maybe even wrong), needs attention.

Reconstruction

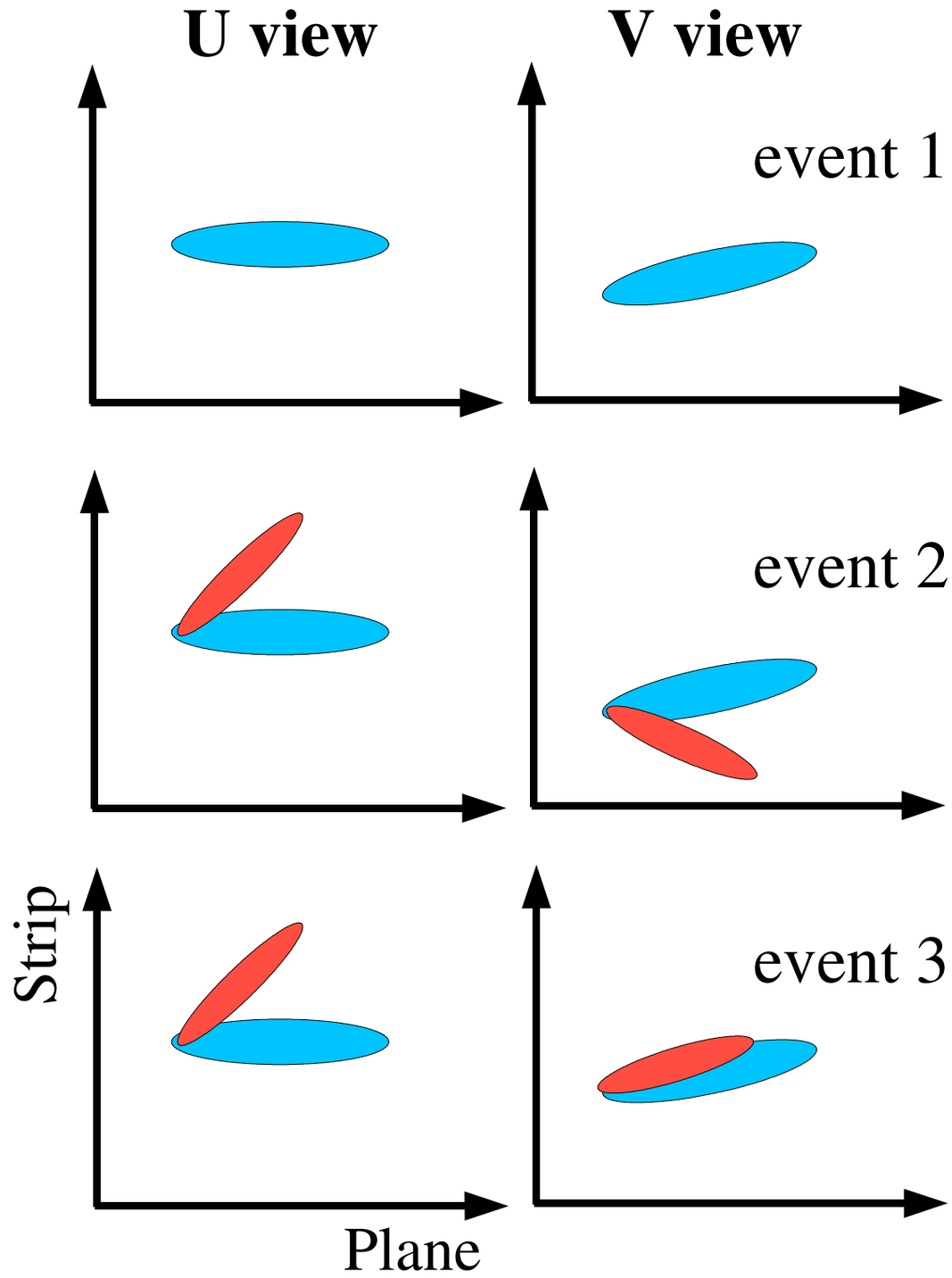
- **Adaption of an algorithm, developed by Hai, that has been used in MINOS with great success, the "SubShower" package.**

- **Looks for shower features in each view (2D shower finding) then combines the views to get 3D showers.**

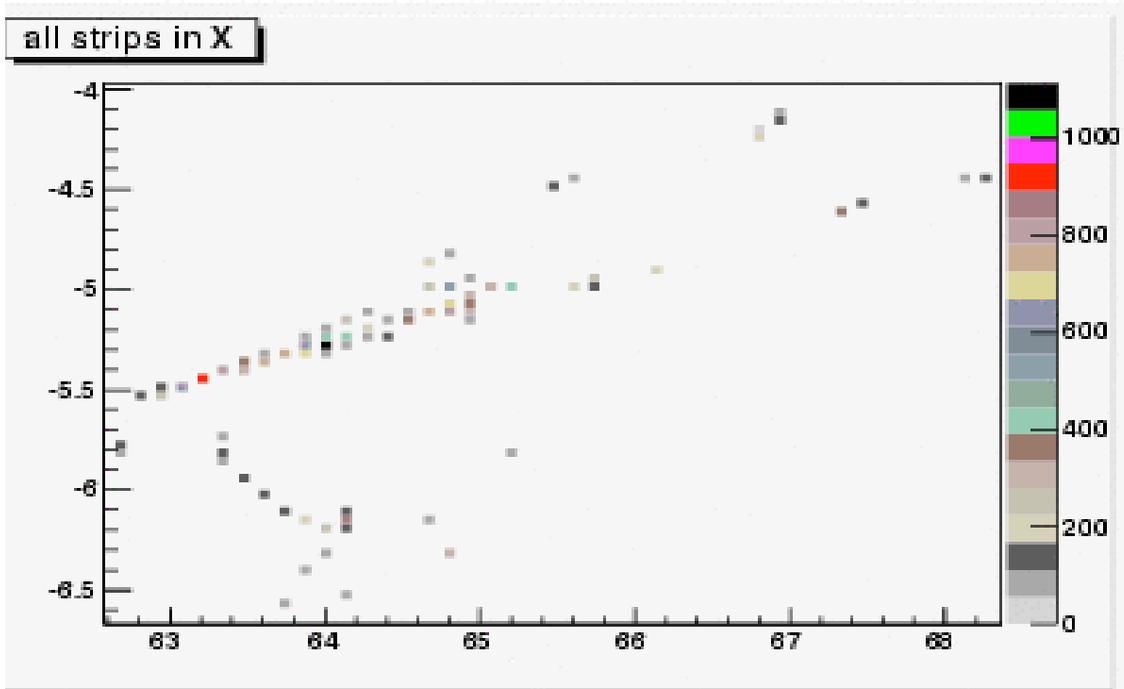
- **Lots of work was done to retune the algorithms for NOvA events.**

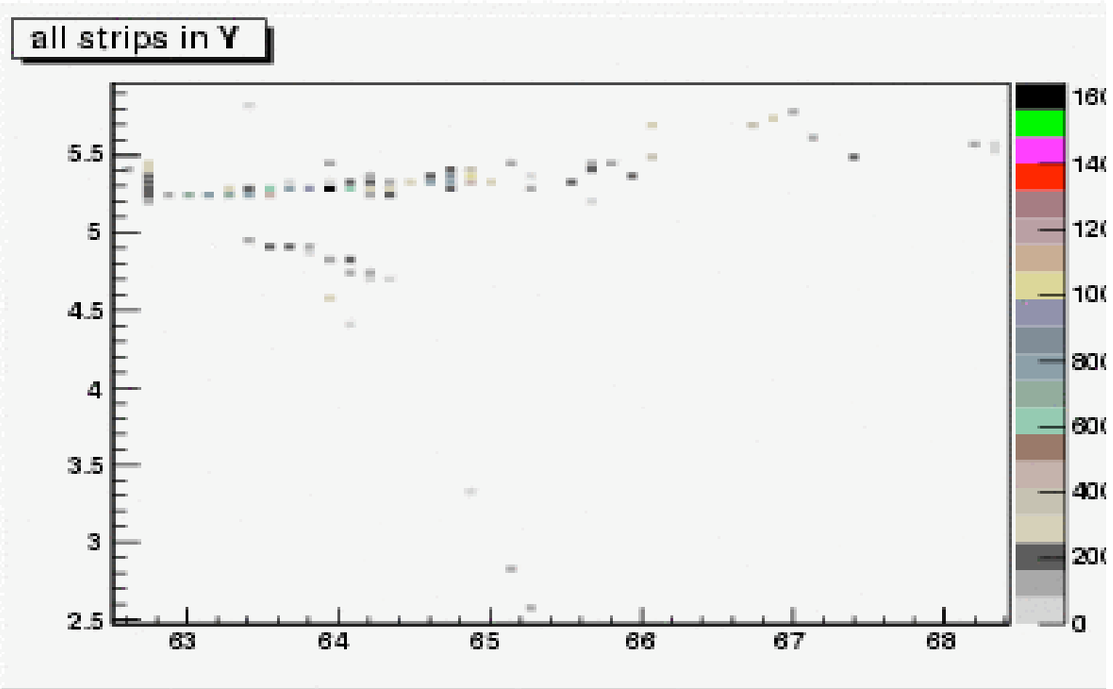
- **Currently we have 2D shower finding working and are working on combining the results.**

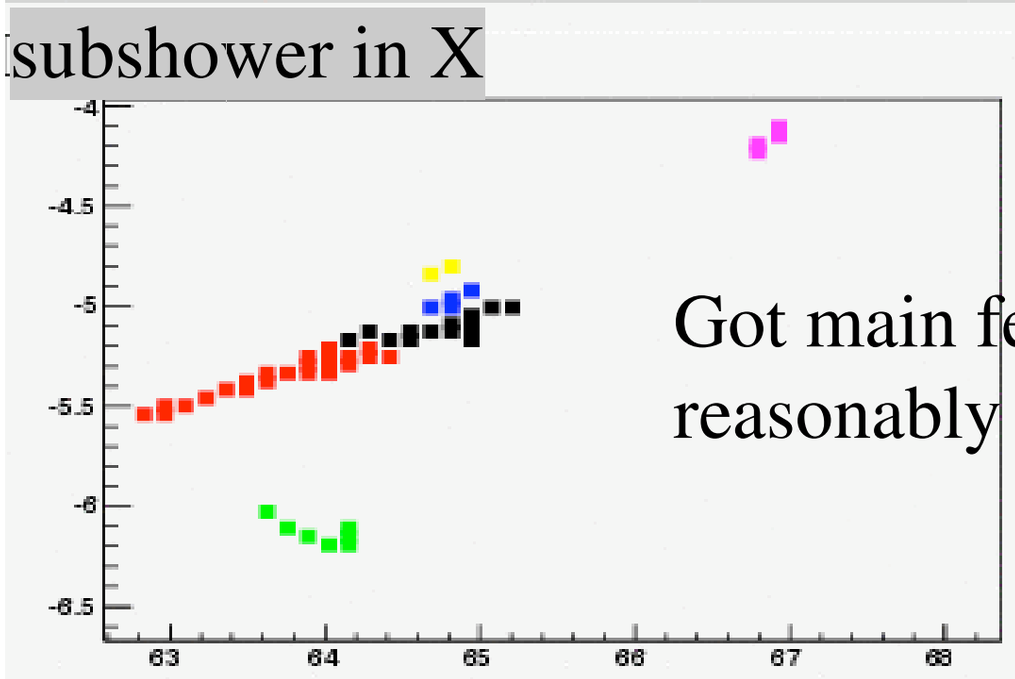
- **Code is in CVS.**



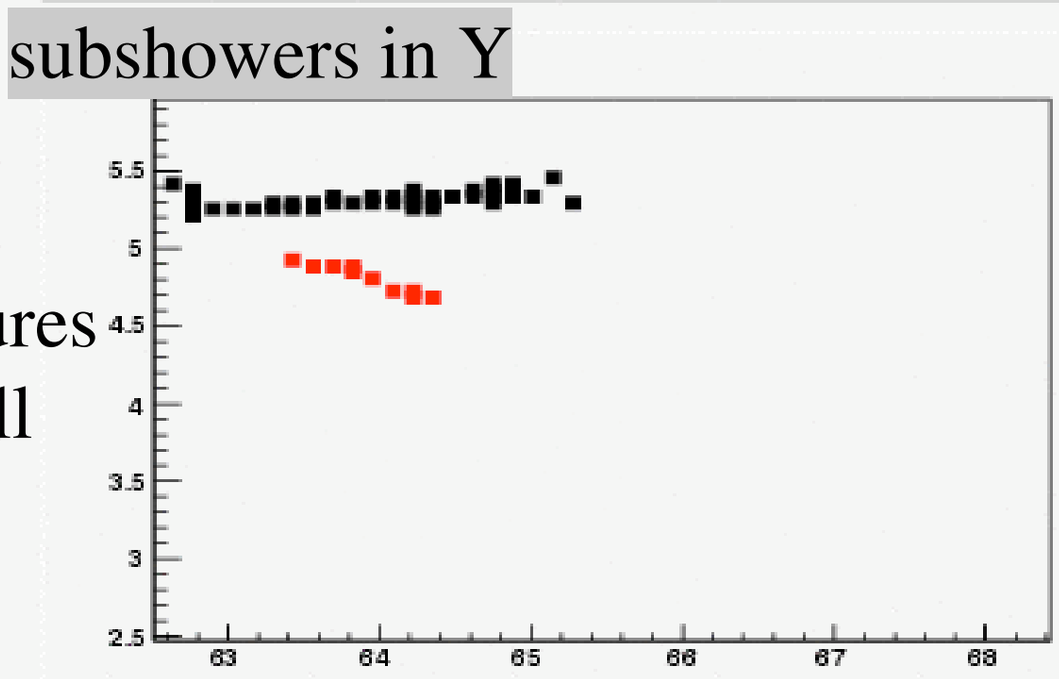
Example Nue Event (tavc)





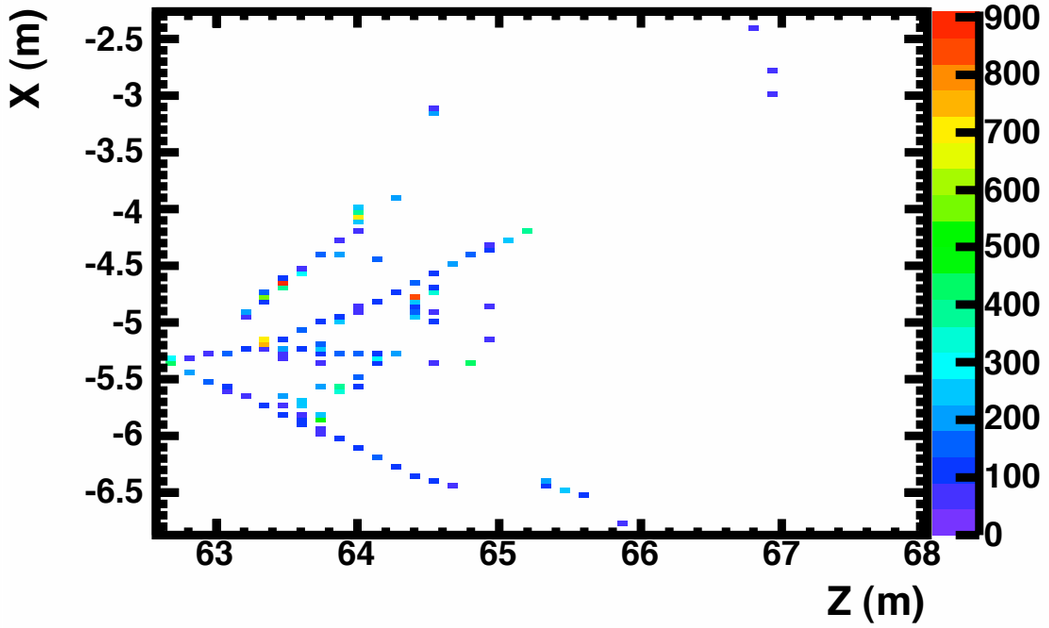


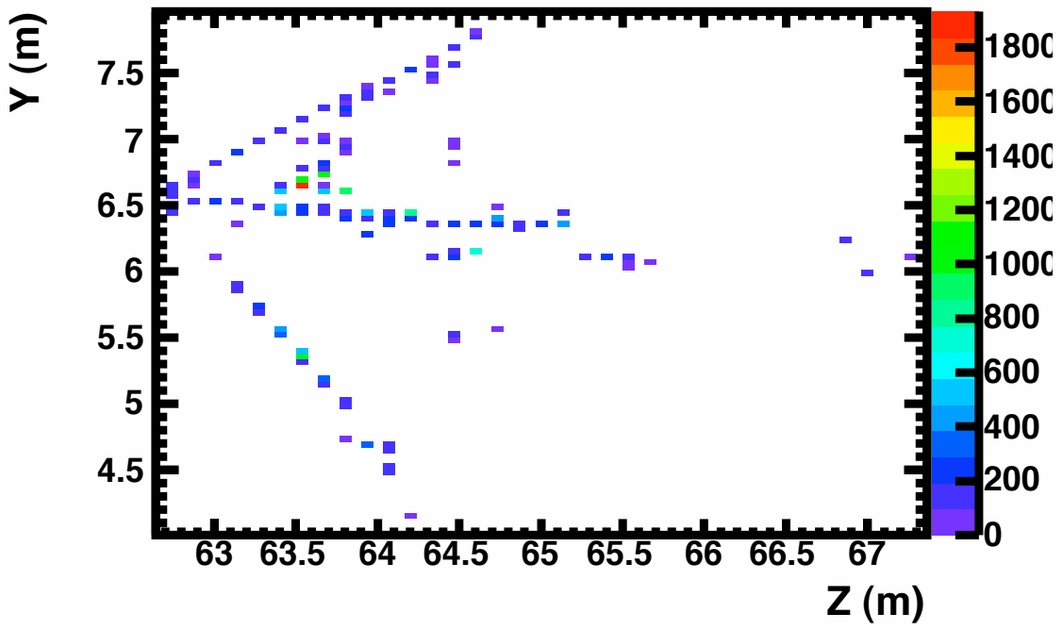
Got main features reasonably well



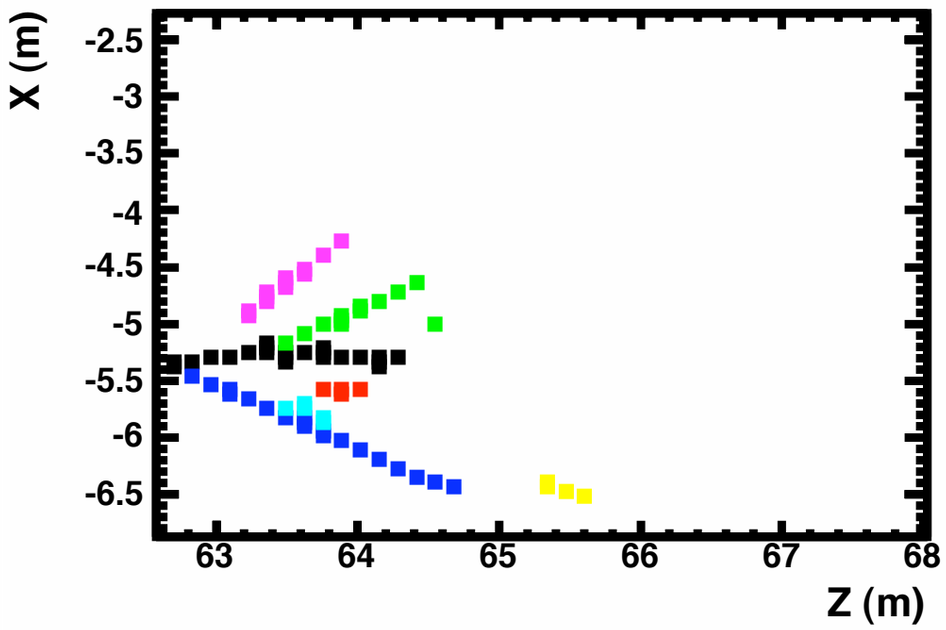
High Mult. Nue Events

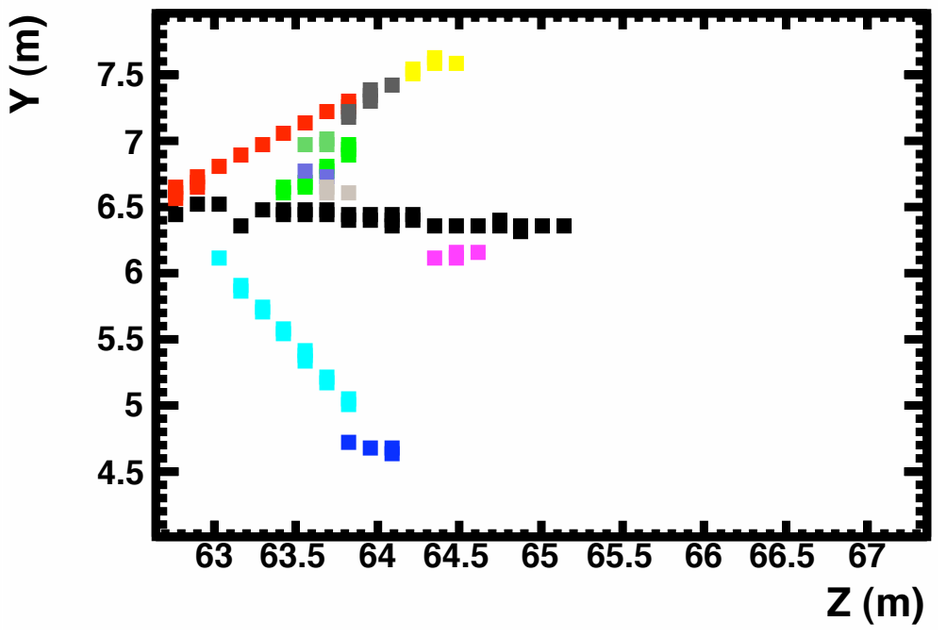
"Raw Data"





Sub Showers

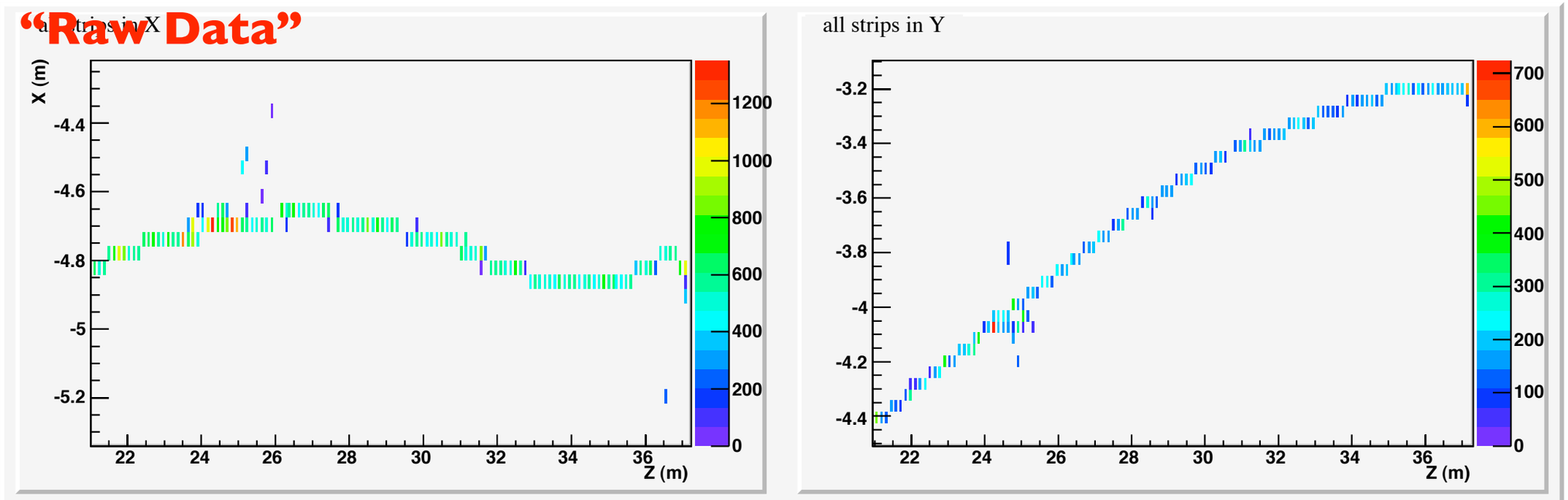




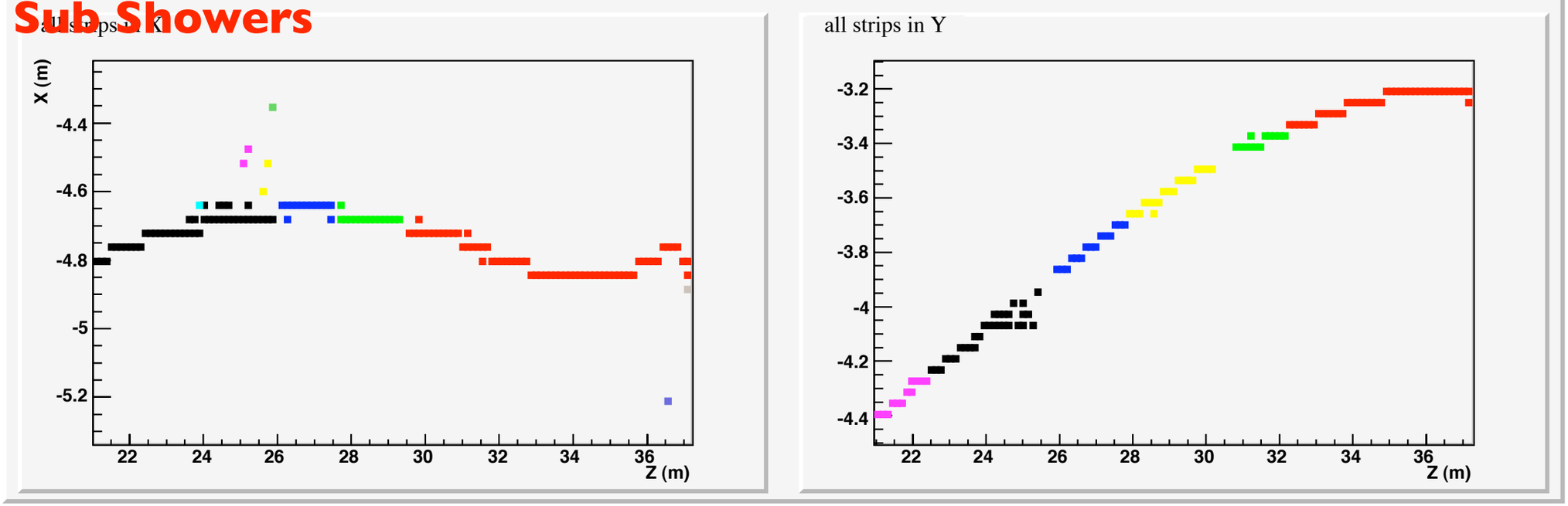
ν_μCC events

Not suitable for nu mu charged current events.

"Raw Data"



Sub Showers



Need some preselection

Remaining Issues

Technical

- Still some issues with CDR geometry (some assumption about alternting planes is still builtin tot he code somewhere)

- Some issues with stack size

Physics

- Working on combining views.

- Then we need to start doing physics:
  - PID,
  - Event Building

Getting and Using SoCal

Supported Platforms

- All flavors of Linux (x86 only).
- Mac OS X 10.3
- Mac Os X 10.4 (PPC and Intel)
- FreeBSD x86 (not fully tested)

FreeBSD®

Dependancies

- GCC 3.0 + (2.95 will probably work too)
- ROOT (sorry) V4.02 and above
- Will keep dependencies to a minimum

Getting + compiling

- SoCal lives in the NOvA CVS (see Brain for access)
- To build go to the SoCal directory and type gmake.

Documentation

- Users guide exists and is kept up to date (at the cost of perfect English)
- Can be compiled from tex source in the socal/docs dir or a PDF can be found on docdb.

Summary

Framework

1. SoCal Framework has been written.
2. While it is not the ultimate framework, it is usable for people to start developing algorithms now. It is fast and simple.
3. Includes all the tools you need to start writing physics code.
4. Works will all detectors and geometries, (including non-ta detectors)
5. Available now... and is documented.
6. Still some “Frills” missing:
    1. DB interface for calibration
    2. Better user experience (e.g. Job Control/Configuration)

Detector Simulation

1. Simple digitization framework developed.
2. The “Dumb” PhotonTransport and ReadoutSim are availble.
3. “Simple” PhotonTransport is on its way
4. No “Simple” ReadoutSim... need help

Reconstruction:

1. Work in progress. Looks very promising.
2. Some issues remaining
3. Help wanted.
4. No tracking at all.